WHAT IS CLAIMED IS:

1. A method for preventing overrun of an input data buffer within a program having the input data buffer on a stack data structure, the program executing on a computing system, the method comprising:

pushing all arguments to a function onto the stack data structure;

pushing a return address onto the stack data structure for use in obtaining the memory address for the instruction to be executed upon completion of the function;

pushing onto the stack data structure a security token, the security token comprises a randomly generated data value;

allocating memory locations on the stack data structure for use as local variables within the function;

completing the instructions within the function;

retrieving the security token value from the stack data structure; and

if the retrieved security token value is identical to the randomly generated data value, return from the function using the return address stored on the stack data structure.

2. The method according to claim 1, wherein the method further comprises aborting the operation of the program if the retrieved security token value is not identical to the randomly generated data value.

3. The method according to claim 1, wherein the randomly generated data value is determined using a random number generator once each time the program is executed.

4. The method according to claim 3, wherein the random number generator generates the randomly generated data value using a snapshot of a system clock within the computing system obtained before the program first accepts input data.

5. The method according to claim 1, wherein the function comprises a subroutine that does not return a data value.

6. The method according to claim 1, wherein the function comprises a subroutine that does returns one or more data values.

7. An apparatus for preventing overrun of an input data buffer within a program having the input data buffer on a stack data structure, the program, the apparatus comprising:

a function call module placing arguments to a function and a return address onto the stack data structure;

a push security token module placing onto the stack data structure a security token, the security token comprises a randomly generated data value;

a perform function module performing the operations within the function, the perform function module allocates memory locations on the stack data structure for use as the input data buffer;

a pop security token module retrieving the security token from the stack data structure upon completion of the operation of the perform function module.

a test module comparing the retrieved security token with the randomly generated data value; and

a complete function module completing the operation of the function;

wherein the complete function module returns from the function if the retrieved security token is determined to be identical to the randomly generated data value by the test module.

8. The apparatus according to claim 7, wherein the complete function module aborts the operation of the program if the retrieved security token is determined not to be identical to the randomly generated data value by the test module.

9. The apparatus according to claim 8, wherein the randomly generated data value is determined using a random number generator module once each time the program is executed.

10. The apparatus according to claim 9, wherein the random number generator module generates the randomly generated data value using a snapshot of a system clock within the computing system obtained before the program first accepts input data.

11. The apparatus according to claim 9, wherein the function comprises a subroutine that does not return a data value.

12. The apparatus according to claim 9, wherein the function comprises a subroutine that does returns one or more data values.

13. A computer program product readable by a computing system and encoding a set of computer instructions for preventing overrun of an input data buffer within a program having

the input data buffer on a stack data structure, the program executing on a computing system, the method comprising:

pushing a return address onto the stack data structure for use in obtaining the memory address for the instruction to be executed upon completion of the function;

pushing onto the stack data structure a security token, the security token comprises a randomly generated data value;

completing the instructions within the function;

retrieving the security token value from the stack data structure;

if the retrieved security token value is identical to the randomly generated data value, return from the function using the return address stored on the stack data structure.

14. The computer program product according to claim 13, wherein the method further comprises aborting the operation of the program if the retrieved security token value is not identical to the randomly generated data value.

15. The computer program product according to claim 13, wherein the randomly generated data value is determined using a random number generator once each time the program is executed.

16. The computer program product according to claim 15, wherein the random number generator generates the randomly generated data value using a snapshot of a system clock within the computing system obtained before the program first accepts input data.

17. The method according to claim 13, wherein the function comprises a subroutine that does not return a data value.

18. The method according to claim 13, wherein the function comprises a subroutine that does returns one or more data values.

19. A computer program product, according to claim 13, wherein the computer data product comprises a set of computer instructions encoded and stored onto a computer-readable storage medium.

20. A computer program product, according to claim 13, wherein the computer data product comprises a set of computer instructions encoded within a carrier wave for transmission between computing systems.